MECHANICAL INTELLIGENCE: RESEARCH AND APPLICATIONS

By: Kurt Konolige, Systems Programmer

Co-principal Investigators
  Earl D. Sacerdoti, Associate Director
  Daniel Sagalowicz, Computer Scientist
  (415) 326-6200   x4840

  Artificial Intelligence Center
  Computer Science and Technology Division

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-1246

MECHANICAL INTELLIGENCE: RESEARCH AND APPLICATIONS

Second Semiannual Report
Covering the Period April 1, 1979 to September 30, 1979

SRI Project 7910
Contract N00039-79-C-0118
ARPA Order No. 3175.28

December 19, 1979

By: Kurt Konolige, Systems Programmer

Co-principal Investigators
Earl D. Sacerdoti, Associate Director
Daniel Sagalowicz, Computer Scientist
(415) 326-6200   x4840

Artificial Intelligence Center
Computer Science and Technology Division

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A/1	

The views and conclusions contained in this document are those of the authors and should not be interpreted as representative of the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

Approved:

Peter E. Hart, Director
Artificial Intelligence Center

David H. Brandin, Executive Director
Computer Science and Technology Division

CONTENTS

ILLUSTRATIONS

iii

I INTRODUCTION

A. Overview

Since October 1, 1978 SRI International has performed research and development in support of the DARPA/IPTO Advanced Command and Control Architectural Testbed (ACCAT) program, under Contract N00039-79-C-0118 with the Naval Electronic Systems Command. Our work has focused on three areas: comprehension of English-language queries about information in a data base, intelligent data-base retrieval, and the representation of knowledge within a computer's memory. We are simultaneously constructing high-performance systems that run reliably in real time and experimenting with and developing more advanced systems that possess extended capabilities but slower response characteristics.

Supported by a previous contract, a milestone version of the system called LADDER (Language Access to Distributed Data with Error Recovery), was conveyed to the ACCAT facility at the Naval Ocean Systems Center (NOSC) in January 1977. As new features have been checked out and proved viable, interim versions of LADDER with increasing levels of sophistication have been installed in the ACCAT facility. The most recent milestone system installation was in August 1979. A subsequent version of the system, as specified in the current contract, will be installed in the ACCAT facility in March 1980. We anticipate that progress during the current research effort will facilitate development of higher performance systems and provide supporting data for evaluation of the long-term role of mechanical intelligence in the ACCAT program. In particular, our current research as presented in the next sections will establish the basic technology for systems whose data bases can be accessed by natural language and which can easily be transfered from one data base domain to another.

1

B. Major Accomplishments

a. Standard LADDER System

LADDER, our complete system for answering English-language queries about information in multiple remote data bases, is being continually updated on the ACCAT facility. The system incorporates a natural-language front end, the SODA program for intelligent data-base access, and the current version of the File Access Manager. The system includes an interface with the situation display program developed at the Information Science Institute of the University of Southern California [11].

b. Recent Accomplishments

During the past six months we have performed a number of studies; listed below are ideas we implemented:

* We implemented a facility that allows the user to refer to ships, ship classes, ports, and commanders even if they have never been defined in the LADDER lexicon. The pattern that the user must employ has the format: <ITEM> NAMED "PROPER-NOUN," such as in the queries   WHERE IS THE SHIP NAMED "TOTOR"? WHICH SHIPS BELONG TO THE SHIP CLASSES NAMED "CLASS1"? WHAT SHIP IS COMMANDED BY A COMMANDER NAMED "JOHN DOE"?

* We recently made several additions to the standard LADDER system to enable its use in conjunction with the multi-level security GUARD system being developed by Logicon. The following is a summary of those additions:

    (1) It is now possible to set a flag (LADDER.ALWAYSPRINT#) so that all answers are always produced. Previously, LADDER would tell the user that a large number of answers are available for his examination and ask whether they should be printed. Setting the above flag will eliminate the question to the user.

    (2) A new function, GUARD, was implemented. It takes one argument—a file name. That file is expected to contain one or more LADDER queries. GUARD processes the queries sequentially. It prints each one, followed by the normal printing made by LADDER. In essence, LADDER operates in a batch mode when used via the GUARD function. When file processing is finished, LADDER prints ***GUARD-END***.

2

* We are developing an interactive facility for defining a new data base for the data access part of our system, SODA. An initial implementation of this program now exists and we are working on improvements to its user interface. The program was released in August 1979 for installation in the ACCAT facility.

* We improved the demonstration system. For example, it used to be the case that "define (X) like (where is the Kennedy)" did not work correctly. If later on, the user typed "X," the system would not access the data base at all. This problem has now been remedied.

* Finally, we improved our interface to the situation display [11]. It is now possible to control the hatching of land masses, the display of highlighting information, and the display of the speed vectors attached to each ship, all in natural language. For a more complete description of those commands, see [6].

Progress continues on D-LADDER, our new system based on the DIAMOND parser. D-LADDER provides the basis for a portable system, in which a good part of the knowledge required for understanding questions is already available to the interface designer who is interested in transferring the natural-language interface to a new application. The next section presents in detail the current status of this work.

II A FRAMEWORK FOR A PORTABLE NATURAL-LANGUAGE
INTERFACE TO DATA BASES

A. Introduction

Despite the success of a number of artificial intelligence (AI) systems as natural-language interfaces to large data bases, the goal of a portable interface remains elusive. We define a portable interface, as a system for understanding natural-language data-base questions that can be easily adapted to a new data base, rather than being "hard-wired" to the data base for which it was developed. We intuit that much of the processing involved in understanding natural-language questions is the same no matter what the form of a particular application's data base. An interface must use many types of knowledge to understand questions: linguistic knowledge about the way words combine to produce meaning, knowledge of the domain of discourse to interpret the question, and knowledge of the structure of the data base to extract relevant data for a reply. Each area of knowledge has a portion that is independent of any particular interface application. For example, it can be argued that much of an interface's knowledge of English syntax is independent of the domain of discourse under consideration.

In this report we will identify application-independent portions of an interface and integrates them with application-dependent portions into a coherent framework. We hope the framework will provide the basis for a portable interface, in which a good part of the knowledge required for understanding questions is already available to the interface designer who is interested in transferring the interface to a new application. To a large extent, the division of the framework into application-dependent and independent portions will be a product of our definition of transportability. The division of the linguistic component, for example, will be different if we permit applications to use only English, or to use different languages. In the latter

4

Additional keywords: LADDER (Language Access to Distributed Data with Error Recovery); man computer interface; ACCAT (Advanced Command and Control Architectural Test bed).

case, the amount of linguistic knowledge common to all applications is a good deal less than the former.

The set of applications we envisage for this framework has the following characteristics:

(1) The natural language is a subset of English.
(2) The domain of discourse is not restricted.
(3) The data base is not restricted.

Assumptions (2) and (3) mean that the framework must be able to handle a very large number of applications -- any application, in fact, in which data are represented explicitly in some data base. The decoupling of the interface framework from a particular data base design and discourse domain has important consequences for the structure of the framework.

A natural-language interface for data base access is portable if it can be transferred with some minimum effort from the data base for which it was originally designed to a new data base. The degree of portability is reflected in the amount of effort necessary to transfer the interface. If an interface is "hard-wired" to a particular data base, major reprogramming will be required to convert it on a new data base. In the extreme case, this effort will be equal to the programming task for the interface to the original data base. On the other hand, an interface that requires no effort to transfer is probably beyond the capabilities of current AI research. Such an interface would have to learn by itself the characteristics of a new data base, and adapt its linguistic and computational abilities accordingly.

What we seek is a design framework that is a compromise between these two extremes of portability that is within the reach of current AI capabilities. An interface that follows the guidelines of this framework would still require some effort to transfer to a new data base, but it would relieve the interface designer of a large portion of the burden of conversion.

So far we have talked rather abstractly about the effort required to transfer an interface. We now concentrate on our LADDER systems [9] to explain the problem areas. S-LADDER is our current demonstration system. It is a state-of-the-art AI system for natural-language access to a Navy command and control data base. Examining deficiencies of S-LADDER from a portability

5

standpoint raised the issues discussed in the next section, and led to this framework for a more portable interface.

B. Types of Transfers

1. Data Model Transfers

The simplest transfer involves moving from a data base with a certain type of data model to a similar data base that uses a different data model. Data model refers to the access method for data [3]; competing access methods can be broadly classed as relational, hierarchical, or network. In general, the query language for data access will be different in the two data bases. We assume that the information they contain is the same but is stored in a different access form.

S-LADDER passes this portability test by providing a uniform relational view of the data, no matter what the underlying data base access method. The query language used by the interface is SODA, a high-level relational language [13]. Special programs convert a query generated in SODA into queries suitable for the real data base underlying the interface. These programs need be written only once for each new type of data base and data base query language. This approach has proven successful in transferring S-LADDER from its original implementation on a relational DBMS to a network DBMS [4].

The strategy that S-LADDER uses is an obvious one, and could be incorporated into any natural-language interface whose goal is portability. The target query language for the interface should be general enough to subsume the query languages of the real data bases for which the interface is intended, yet simple enough so the translation will not be too difficult. SODA fulfills these requirements.*

---

*The issue of efficiency of access can be complicated if this strategy is used; queries that are efficient in one data model may not be in another: see [13].

6

2. Logical Organization Transfers

The logical organization of the data base is the way in which the interface views the information it contains. The second transfer problem is to take an interface designed for a data base with a certain logical organization and move it to a data base containing identical information, but with a different logical organization. The more tightly an interface is linked to a particular data base's logical organization, the more difficult this transfer will be.

A simple example will illustrate how two different logical organizations can be equivalent in information content.* DB_1 has the following file:

SHIPS(NAME,COUNTRY)

where NAME is the name of the ship, and COUNTRY is the country that owns it. Each record in the file has the name and owner of a single ship. DB_2 has a slightly different organization:

US-SHIPS(NAME)

FOREIGN-SHIPS(NAME,COUNTRY)

Each record in the US-SHIPS file has the name of a ship owned by the U.S.; foreign ships are assigned records in the FOREIGN-SHIPS file.

These two different organizations obviously can contain exactly the same information, in the sense that any set of facts that can be represented in DB_1 can also be represented in DB_2. Yet the logical structure of the data bases is different, and this has important consequences for the queries written to extract information from them. For example, a query to count the number of U.S. ships would have a restriction on the COUNTRY field in DB_1, but no restriction in DB_2, since the ownership information is implicit in the file US-SHIPS.

S-LADDER does not pass this transfer test as readily as it did the

---

*For simplicity, we assume from here on that a relational data model is being used.

7

previous one. Although some changes in data base organization can be absorbed with little change to S-LADDER, there are some subtle interactions with S-LADDER's assumptions about the way data is stored in a data base. A good *example of this occurs frequently with the* COUNT operation. Consider the natural-language question

"How many countries own ships?"

For the logical organization of DB_1, the correct query would be to simply count the unique COUNTRY fields in the SHIPS file. For DB_2, the query is not so simple; all the unique fields in the FOREIGN-SHIPS file must be counted, and this count incremented by one if the US-SHIPS file is nonempty. Transferring from DB_1 to DB_2 would require major changes in the way S-LADDER translates certain queries.

The difficulty in transferring S-LADDER stems from a design assumption that this interface makes about the logical organization of its underlying data base. S-LADDER assumes there will be a simple correspondence between the objects to which natural-language questions refer and data base files, and between attributes of these objects and fields in the files. DB_1 is a much more suitable data base for S-LADDER, because ships, a frequently occurring object in natural-language questions for this domain, are represented simply in a single file with one record per ship.

In any natural-language interface that attempts to insulate the user from knowing the exact logical organization of the data base, there will probably be a fundamental difference between the user's model of the domain and the *encoding of the domain in the logical organization of the data base.* Where the interface designer has control over the data base organization, he can attempt to model the user's view of the domain as closely as possible in the organization of the data base, and so, for example, he would choose DB_1 over DB_2. However, the transfer problem obviates this solution, since the interface must in principle be transferable to data bases with widely varying

8

organizations that may not parallel the user's domain view.*

Rather than use the data base's logical organization as the target representation for the interface, we can consider using a representation that encodes the user's model of the domain; we call this representation a conceptual schema. In its simplest form, a conceptual schema must encode the user's knowledge of the objects in the domain and the relationships between them. Since the conceptual schema remains unchanged in switching between data bases that contain the same information, translation of natural-language questions into a conceptual schema representation requires no reprogramming at this level. Translating from the conceptual schema into the correct query with respect to the data base's logical organization can still be a difficult problem, though. By interposing a conceptual schema between the natural-language interface and the data base's logical organization, this problem is isolated from the task of natural-language translation. Then, transferring between data bases in the same domain with different logical organizations involves changing only the translation from conceptual schema to the real data base query, rather than reprogramming the entire natural-language translation.

While the primary motivation for including a conceptual schema arose from considerations of transportability, there are several additional reasons why it makes the task of interpreting questions easier and provides some capabilities that could not be accommodated by direct data base translation alone. One reason for this is that it allows for greater conceptual coverage. A data base typically will have information only about some portion of a given domain; the user may not be aware that information is missing. Consider the questions:

    How many British ships have doctors on board?
    How many U.S. ships have doctors on board?
    How many ships have doctors on board?

_____

*[10] and [15] are examples of interfaces in which the designer has full control over the logical organization of the data base. To the extent that the data base's organization can capture the user's domain model over a series of representative questions, these systems will be successful. However, as is argued subsequently, a data base schema alone does not comprise an adequate model of the domain for questions that demand even trivial inferences about the domain for their interpretation.

9

The user would expect a competent interface to understand all three queries, if it understood any one of them. Yet an interface that depends on a data base schema for its model of the domain may very well interpret one of the queries correctly, while either failing to interpret the others, or giving wrong answers to them. For example, suppose one of the data base files contained information about the doctors on board U.S. ships only:

US-SHIPS(NAME,MEDICALCAP)

where NAME is the ship identifier, and MEDICALCAP is true if there are doctors on the ship. It is part of more general knowledge of this domain that ships of nations other than the U.S. can have doctors on board. In light of this, the answers to the first and third questions should be:

How many British ...?   I don't know.

How many ships ...?   I only know about U.S. ships: there are
                       x of them with doctors on board.

The data base schema is simply a collection of files, fields within these files, and a description of the functional relationships between these fields. Having just the US-SHIPS file, the interface cannot distinguish the case of the data base not having information about foreign ship doctors, from the case of it being true (in the domain) that foreign ships don't carry doctors. It is impossible to interpret the user's question correctly without this additional information.

Of course, it is always possible to represent more general world knowledge in the data base. The knowledge about what nations' ships could have doctors could be encoded in the file:

HAVE-DOCTORS(NATIONALITY)

where there would be one entry for each country whose ships could have doctors on board.* But data base designers typically do not insert such

---

*We may imagine, for example, that country X has a shortage of doctors, and so refuses to put them on ships when they are needed more urgently at home. Country X would not be listed in the HAVE-DOCTORS file.

10

information into their data base, precisely because it is general knowledge about the domain that any human question-answerer would be expected to have. Further, data bases that are designed to store large amounts of data are not good repositories for general knowledge about the domain because they are not structured to permit efficient inferencing about the domain. Many AI attempts to construct efficient and economical representations of domain knowledge attest to the difficulty of this representation problem. The conceptual schema uses AI representation techniques to encode a more general world model of the domain than is economically possible with the data base schema.

The advantage of a conceptual schema, then, is that it can economically encode more information about the domain of discourse; inference processes that use this knowledge are an important part of an interface that attempts to understand natural-language questions with any degree of fluency. In section II.C.1 we will present a more detailed example of processes that use linguistic clues from a question to guide inferencing about the domain as encoded in a conceptual schema. The point is not to show the exact nature of the processes, but to indicate by example the need for a more comprehensive representation of the domain than is availbable from a typical data base schema.

There are other facets of natural-language question-answering dialogues not captured by a representation of the domain. We might ask, for example, how knowledge of the user's goals and beliefs should influence the interpretation of his questions.* These facets of question-answering argue even more for some represntation that is separate from the bulk data storage of the data base. We will group all issues of inference and representation of the communication process at the conceptual schema level and have more to say about them with respect to transportability in section II.B.4.

_____

*See [5], for example, for a discussion of some of these issues, and examples of systems that attempt to take them into account. Although these systems are not concerned with data base access per se, they do address the problem of information retrieval in general.

11

3. Domain Transfers

The most difficult transfer task is between data bases of different domains. Most linguistic and semantic interface capabilities tend to be oriented toward a particular domain; changing domains involves major reprogramming of domain-specific coding. We examine linguistic and semantic capabilities below.

a. Linguistic Coverage

In many natural-language interfaces there is no separation between linguistic knowledge and domain knowledge. A common technique in these interfaces is the use of a semantic grammar: productions in the grammar are phrased in terms of domain concepts like ships, rather than linguistically motivated syntactic categories like noun phrase. A typical rule in S-LADDER's grammar is:

Do <SHIPS> <HAVE> <SHIP-ATTRIBUTE>   (R1)

where do is a terminal symbol, and <SHIPS>, <HAVE>, and <SHIP-ATTRIBUTE> are metasymbols that can be expanded by other grammar rules.

Semantic grammars attach domain semantics very closely to the grammar rules, and consequently it is very easy to write and interpret these rules for a set of typical natural-language questions in a domain. On the other hand, such rules lack linguistic generality. If we want to recognize the passive variation of rule R1 above, we must write a new rule for it, and similarly for every rule written in the active form. There is no easy way to capture the linguistic generalization of the passive transformation in a semantic grammar.

When transferring an interface with a semantic grammar between different domains, the entire grammar must in general be rewritten; there is very little linguistic knowledge that is independent of the domain. Referring to the passive variant, in the new domain the interface designer must remember to write the passive variation to every new active grammar rule he writes.

Besides the difficulty of domain transfer, another consequence of the lack of linguistic generalization is the difficulty of achieving any measure of linguistic coverage using a semantic grammar. A set of semantic rules that capture the typical questions of a domain may not account for even simple

12

linguistic variations in their phrasing. Such variations can be accepted by writing more semantic grammar rules, of course, but this leads to a potentially combinatorial increase in the number of rules required to achieve even moderate linguistic coverage.

To achieve transportability between domains and linguistic coverage within a domain, a separate level of linguistic knowledge is called for. This level must have a linguistically motivated grammar if it is to capture linguistic generalizations, such as the passive variant of active sentences, that are common across all domains. This level achieves transportability precisely because its rules are valid for the language as a whole, rather than for the fragment applicable to a particular domain. The level achieves coverage because we can concentrate on encoding a large subset of English syntax within the grammar; any domain to which it is applied will then have that fragment available to it.

Having a general, linguistically motivated grammar does not solve the transfer problem entirely, however. There will always be some syntactic constructions that are peculiar to a given domain (e.g., idioms) and cannot be encoded in a general way. A great deal of lexical information is also domain-specific; the interface designer must specify the lexicon for a domain.* Finally, there is the hard task of integrating linguistic knowledge with domain knowledge. We turn next to a method for accomplishing this integration in a flexible manner.

b. Basic Semantic Operators

Let us examine a typical rule for active sentences that a linguistically motivated grammar might have:

S -> NP VP   (R2)

where S is a sentence metasymbol, NP and VP are noun phrase and verb phrase

---

*A number of words will be common across many domains, and can serve as a core lexicon. Typically, purely functional words will be in this lexicon, e.g., "is", "have", auxiliary verbs, wh-words, etc.

metasymbols. Additional restrictions on this rule provide for number agreement between NP and VP, account for the correct number of arguments to the verb, and the like. These checks are independent of the domain, and hence this rule gives only a partial indication of whether a sentence is acceptable or not. In the ships domain, both

>"The officer commands the ship"

and

>"The ocean commands the ship"

are accepted by Rule R2; only domain semantics can reject the second one. Further, simple acceptance of the first sentence is not enough; we need to build a representation of the meaning of the sentence in the domain.

To accomplish these goals, we introduce the concept of <u>basic semantic operators</u>, which are a set of functions that when evaluated build semantic structures in the domain. The semantic operators provide a flexible, uniform interface between linguistic analysis of a sentence and domain semantics. Sentences with similar meaning but differing phrase structures are mapped into similar sets of semantic operators. For example, consider the simple passive transformation that relates the two sentences:

>Who commands the Kennedy?

>By whom is the Kennedy commanded?

Even though the surface syntactic structure varies, the propositional structure of the sentences is the same: a predicate, <u>commands</u>, with two arguments, <u>the Kennedy</u> and <u>who</u>. Both sentences would invoke the same semantic operators to construct a representation of the propositional content. The translation from surface structure to calls on semantic operators thus encodes linguistic knowledge about the way surface structure relates to meaning.

It is important to note that the translation into calls on semantic operators is domain-independent, that is, it was not designed with a particular subject area in mind. Thus we hope that it will provide a uniform way of analyzing the syntactic structure of sentences, relieving the natural-language interface builder of a large part of his task.

Currently there are four categories of semantic operators:

14

* Propositional operators -- which are derived from the propositional content of a sentence. They include calls to create sets (corresponding mainly to noun phrases), and propositions relating these sets (such as verb phrases or adjectival modifiers).

* Request operators -- which add various types of request information from the sentence. They include counting ("how much," "how many"), yes/no, and set specification ("what ships are ...").

* Modality operators -- which deal mainly with the tense and mood aspects of verbs.

* Quantification operators -- which deal with the quantificational structure of a sentence, including explicit ("each," "every," "all," and so on) and implicit quantification (e.g., superlatives).

To illustrate the way in which the various types of semantic operators are invoked, consider the question:

When could all the subs in the Med reach Gibraltar?

The propositional structure of this sentence is straightforward: a predicate, reach, that connects three arguments, Gibraltar, subs in the Med, and when, a time argument. In addition, the argument subs in the Med itself has a propositional structure based on the implicit containment predicate in, which relates subs and Med. Thus two calls to the propositional semantic operators are invoked; first to build up the phrase subs in the Med, and then to attach arguments to the predicate reach.

The request operators are invoked for the argument when to indicate that the value of this argument is being requested by the question. The quantifier operators are invoked to attach the quantifier all to the phrase subs in the Med. Finally, the modality operators are invoked to indicate that the modal auxiliary could is attached to the predicate reach.

At this point, the translation process has extracted all the information it can from the sentence, and converted it into a set of calls on the semantic operators. Obviously, there is more information in the sentence that cannot be encoded by these four types of semantic operators. Here we rely on the inherent flexibility of the operators; they can be augmented as our linguistic analysis grows more demanding. For example, if there is some theory of stative versus active interpretation of verb structure we wish to incorporate, we simply define a new class of semantic operators, along with the programs that produce

15

calls to these functions from the phrase structure of a sentence.

The semantic operators also provide a flexible approach to the problems facing an interface builder, since he need use only the operators required by the problem at hand. For example, in designing the interface for a query system, the propositional and request operators would be most important. Operators that dealt with more subtle aspects of the sentence, such as quantification or the modality of verbs, could be added later.

4. A Framework

Figure 1 summarizes the discussions of the previous sections.

Level	Representation	Translation Process	Instantiation
Lexical:	Dictionary*		English Sentence
Syntactic:	Linguistic Grammar	Parser	Annotated Parse Tree
Domain-Independent Semantics:		Translators	Semantic Operator Calls
Domain Semantics:	Conceptual Schema*	Semantic Operator Evaluations*	Conceptual Schema Query
Virtual Data base:	Virtual Data base Schema*	Translation Routines	SODA Query
Real Data base:	Real Data base Schema*	Translation Routines	Real Data base Query

Figure 1: Framework for a Transportable Interface

The interface is divided into six levels based on the type of information

16

represented. The declarative knowledge column lists the declarative form of knowledge used at each level. Note that at the domain-independent semantics level there is no explicit declarative representation, since all information is encoded in the translator functions for this level. The instantiation column lists the structures built at each stage in the translation process, starting from the original sentence. The translator process column specifies the routines that convert instantiations at one level into instantiations at the next, using information in the declarative knowledge column.

The starred items are those parts of the interface that must be redefined for different types of data base transfers. The more difficult transfers involve changing more parts of the framework. Moving to a new domain, for example, means that all the starred items must be rewritten. Note, however, that only one of these items is procedural in nature -- the semantic operators. A great deal of the purely linguistic processing on the first three levels is taken care of, relieving the interface designer of a large burden. In writing the interface for a new domain, he can concentrate on those portions of the interface most concerned with domain semantics.

Even in the most difficult interface transfer--from one domain to another--it may be possible to utilize a significant portion of the starred information in the new domain. The dictionary is divided into domain-independent and domain-dependent parts, as indicated previously; only the latter need be changed in a transfer. Programs that evalute the semantic operators may also be transferable, if the discourse domains have a similar context. For example, if both discourse domains are concerned with information retrieval from a data base, it may be possible to use the same semantic operator evaluations and just change the conceptual schema. If the discourse context changes drastically and introduces as a consequence new elements such as causality of events or beliefs and plans of the user, then the semantic operator evaluations will have to be rewritten.

Most of the framework flexibility comes from the separation of the conceputual level from linguistic processing on the one hand and data base processing on the other. Nothing has been said about the form of the conceptual schema; the interface designer is free to choose any representation (partitioned networks, frames, and so on), and include any processing he thinks

17

appropriate for the domain (such as planning or modeling the user's goals and intentions). To the extent that the semantic operator calls capture the appropriate linguistic clues from the input and transfer them to the domain semantics, we can hope to be successful in decoupling these levels. Experiments will be performed to test this idea soon; currently we have implemented the framework for a single domain.

C.   An Implementation

We have implemented the framework previously described, using the Navy command and control domain of S-LADDER. All parts of the framework have been implemented, and the system is currently producing queries on an actual data base, based on natural-language questions from users. On the domain-independent side of the framework, we have the following:

* A linguistically motivated grammar of a large fragment of English, called DIAGRAM [7].
* A bottom-up parser for that grammar, called DIAMOND [4].
* Translators that produce calls on the semantic operators, given an annotated parse tree.
* Translation routines from a conceptual query language to the SODA query language.
* Translation routines from SODA to two real data base query languages.*

These processes and structures form the core of a transportable interface. To parameterize the interface to D-LADDER's domain required adding a lexicon, a conceptual schema for that domain, and semantic operators to construct the appropriate semantic representation within the domain. The virtual and real data base schemata were taken over directly from S-LADDER. To show how the interface works, we present a short example, concentrating primarily on the interaction between the conceptual schema and the linguistic and data base levels.

---

*Taken directly from S-LADDER.

18

1. An Example

Our implementation of a conceptual schema is based on a network representation of objects in the domain and their relationships. This approach is founded on techniques for knowledge representation developed at SRI over several years [12], although it may also be viewed as being "frame-like" in the sense that a number of recent AI languages are [2] [14]. There are two basic parts to the representation: an object set taxonomy which gives subset and superset relations among various sets in the domain; and delineations, which specify the types of arguments relations can have. In addition, there are some metastructures that control inferencing processes that operate during the interpretation of a question, and which are thus not strictly a part of the domain representation.

Although exact details of the representation are too complicated to report here, we will describe how it is used to actually construct an interpretation of a question. Consider:

Who commands the Lafayettes?

The lexical entries for who and Lafayette have pointers to sets in the domain representation. Figure 2 shows the relevant portion of the taxonomy. who refers to the class of LEGAL-PERSONS; subsets of LEGAL-PERSONS, labeled by s-arcs, are COUNTRIES and OFFICERS (note that one could ask, Who owns the Kennedy, where who refers to a country). Lafayette refers to the individual node LAFAYETTE, which can be an element (e-arc) of either NAVAL-SHIP-CLASSES or NAVAL-SHIPS, i.e., the Lafayette class as opposed to the ship Lafayette. At this point, then, there is ambiguity in the references for both nouns. An important part of the representation is that these ambiguities can be easily accommodated by the taxonomy structure.

The representation also shows what relations can exist between objects in the domain. The delineations for the COMMANDINGS and NAVAL-CLASS-MEMBER relations are shown in Figure 3. The arcs emanating from these nodes are labeled with the argument type, and point to the set from which the argument must be drawn, e.g., the "commander" argument of COMMANDINGS must come from the set OFFICERS.

19
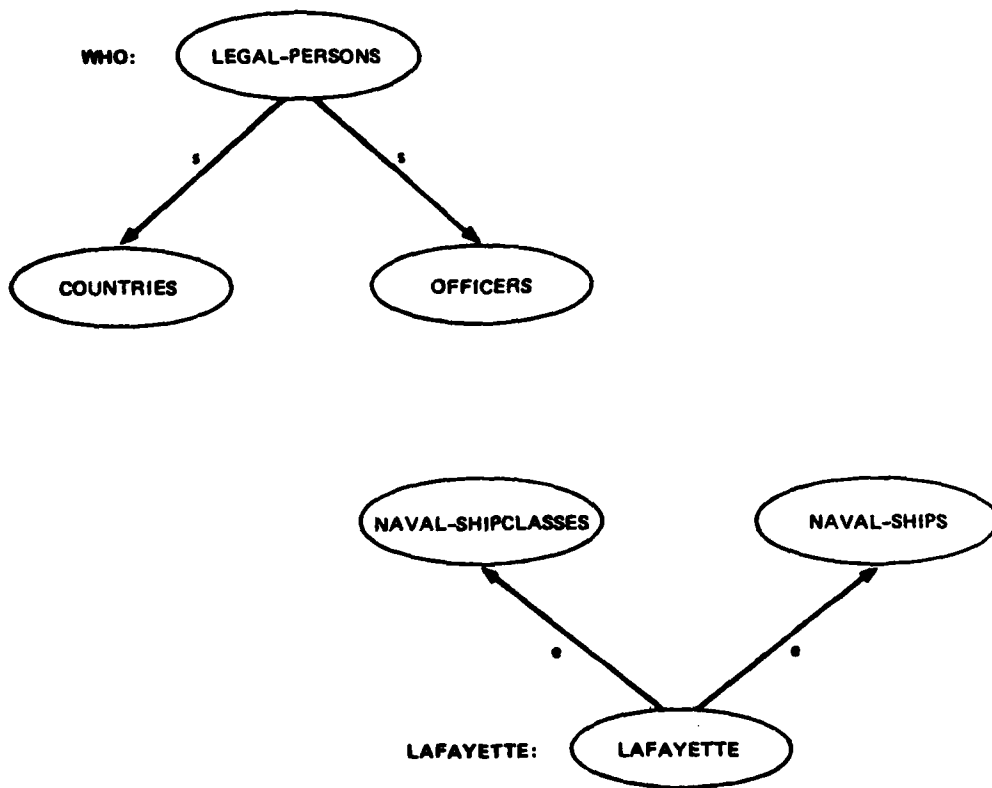
Figure 2: Fragment of the Conceptual Schema Taxonomy
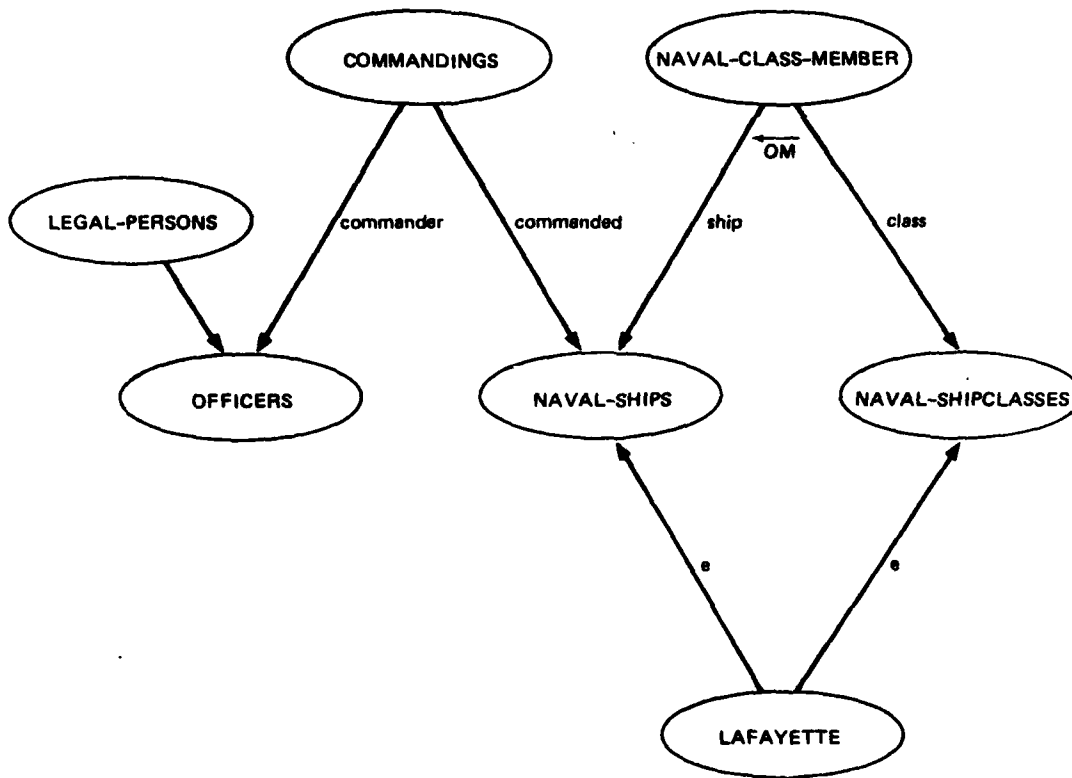
Figure 3: Fragment of the Conceptual Schema Delineations

21

The verb commands in the query refers to the COMMANDINGS relation. Syntactic information from the sentence tells us that who, as subject, must be the "commander" argument. Thus we have identified who as belonging to the OFFICERS subset of LEGAL-PERSONS, rather than COUNTRIES.

Similarly, we find that the Lafayettes must be the "commanded" argument of COMMANDINGS; i.e., it belongs to NAVAL-SHIPS. However, there are additional clues from syntax that force a different interpretation. the Lafayettes is plural, and hence demands a reference in the domain that is not necessarily a single object. Since the LAFAYETTE node interpreted as a NAVAL-SHIPS is a single element, this can't be the correct interpretation. In an alternate interpretation, the node LAFAYETTE is an element of NAVAL-SHIP-CLASSES; by the NAVAL-CLASS-MEMBER relation, a single class can stand for a set of ships, and so is an acceptable interpretation of the plural syntax. Thus the syntactic clue of plurality is used to disambiguate the semantic interpretation of the reference to the node LAFAYETTE. Note that this type of inference is impossible in the S-LADDER system, where many syntactic clues, among them plurality of noun phrases, are simply disregarded by the grammar. To S-LADDER, Who commands Lafayette and Who commands the Lafayettes look exactly alike.

The OM symbol on the NAVAL-CLASS-MEMBER relation is part of the metastructure of the instantiation. It signals that this relation is a special type called an owner-member: each class "owns" a set of ships, and the class can be used to stand for this set. Metastructures are used to control what would otherwise be a large search space of inferences, by classifying relations as special types that can only be used at specific points in the inference process.

2. Generating SODA Queries

Once a question has been interpreted with respect to the conceptual schema, a SODA query must be generated from its conceptual schema representation. Because the way in which data are stored by the data base may not correspond directly to the way the conceptual schema encodes a model of the domain, this task can in general be arbitrarily hard; that is, it may involve an arbitrary amount of inferencing to produce a correct query.

In the absence of a general-purpose inference mechanism to produce

22

SODA queries, we have concentrated on several techniques that promise to cover the most common types of attachments between a data base schema and a conceptual schema. This section describes these techniques, and gives examples of how they can be used to attach our conceptual schema to the schema of the Blue File [1], the Navy's command and control data base.

a.   Simple Predicates

In the best case, a predicate in the conceptual schema will correspond to a relation in the data base schema. This is true for the predicates given in the previous section -- namely, COMMANDINGS and NAVAL-CLASS-MEMBER. The commander of a unit is found in the UNIT relation, and the class of a ship in the SHIP relation. The actual generation from the conceptual schema involves setting up SODA tuple variables for each predicate in the query, and SODA terms for each set.* Where two predicates share a common argument, an equijoin is produced in the query code. Thus in Figure 4 we have the conceptual schema representation of Who commands the Lafayettes, and the SODA query generated, with the origin of the various subexpressions of the SODA query indicated.

Although this example was rather straightforward, various subtleties of the data base design can cause complications. For example, an object in the conceptual domain, like SHIPS, is not always referred to in a consistent manner by the data base. Some files may use the name of the ship, while others use a vehicle identification code that uniquely identifies the ship. The generation process must take these quirks into account.

b.   Functions

In the conceptual schema, functions are treated as a special type of relation that must be handled separately by the translation process, since they have a different SODA syntax from ordinary relations. Also, care must be used to ensure that their arguments are defined before they are invoked. For

_____

*We will assume familiarity with the SODA query language; see [13] for a description.

23
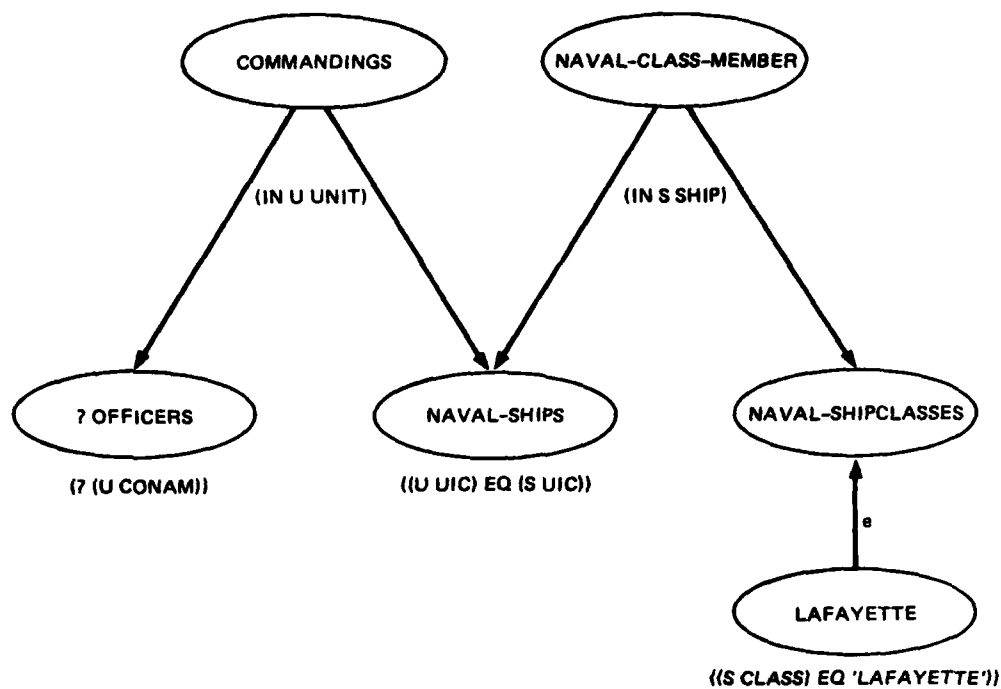
Figure 4: Generation of a SODA Query
from a Conceptual Schema Instantiation

24

example, consider the question:

What positions are 300 miles from the Lafayette?

Such a query could not be answered by the data base because the distance function needs to have both of its position arguments defined. For the above question, the result of the distance function is specified (300 miles), as well as one of its position arguments (the Lafayette); but the function cannot be inverted to generate a list of positions that are 300 miles from the Lafayette. The generation process must recognize this condition and give an appropriate error message.

A second peculiarity of functions is that they may not be executable by the DBMS that contains the data base. In this case, the generation process must send off SODA queries to retrieve the arguments of the function, evaluate the function, and then put the result into another SODA query.

c.   Implicit Sets

Not all sets that the conceptual schema knows about correspond to relations in the data base. For instance, there is a SHIP relation where each tuple gives information about a ship, but there is no TANKER relation. Such conceptual schema sets as TANKERS usually have an implicit definition in the data base, in the sense that we could form a more or less complicated SODA restriction over some relation to extract the set. If we are lucky, there will be a single field whose value will tell us whether a tuple is a member of a restricted set or not, e.g., the TYPE field of the SHIP relation tells us if the ship is a tanker. Because of peculiarities of various data base query languages, the extraction of implicit sets may be impractical. This is because, while most query languages are first-order complete, they may not have operations that allow the efficient extraction of certain subsets of the data. Consider the case in which the implicit set is ships that left port at night, and the data base represents departure times in the form of strings such as "12:30AM" or "2:02PM." An efficient method of extracting this set would be to test if the departure time contained the string "AM" or "PM." Many query languages do not have string operations, however (including SODA). For these languages, it is theoretically possible to extract the set by exhaustively listing all PM times

25

in a disjunction; such a method is clearly impractical, and infeasible for most DBMSs.

d.   Existence

At times a data base field will indicate the existence of an object or property rather than its exact value. A typical example of this is the MED field in the SHIP relation. If the value of this field is D, then there is a doctor on board the ship, but it doesn't give the identity of the doctor. The presence of these existence fields causes problems for generation of data base queries, because it means some questions that ask for the existence of an object will be answerable, while others that ask for the name of the object will not. Consider the questions:

> Is there a doctor on the Biddle?
> What doctor is on the Biddle?
> How many doctors are on the Biddle?

Only the first of these questions can be answered, given the above definition of the MED field.

It should be obvious from the examples that the process of generating a correct SODA query from the conceptual representation can be a complicated one. To take this even one step further, we could imagine trying to come up with a reasonable query when there was no correct way to answer the question directly, e.g., to respond "At least one" if asked "How many doctors are on the Biddle?". It would then seem impossible to generate SODA queries without considering general dialogue issues such as the goals and presuppositions of the user.

D.   Discussion of Results

The current state of implementation has enabled us to evaluate the framework to some extent. We will discuss two issues in this section: the split between linguistic processing and domain-dependent processing implied by the framework, and the goal of linguistic coverage.

1. Linguistic versus Domain-Dependent Processing

The framework shown in Figure 1 has explicit levels for syntax and domain-independent semantics (which we will call the linguistic levels) and domain-dependent semantics. In our straightforward implementation of this framework, processing proceeds sequentially among these levels: first an annotated syntax tree is derived, from this calls to semantic operators are generated, and finally a conceptual schema instantiation is produced. Because in this implementation there is no feedback between levels, all ambiguities are kept at each level. As processing proceeds to higher levels, these ambiguities are weeded out. A scoring algorithm is used to rate interpretations at each level of the framework, and the best scoring interpretation is assumed to be the intended meaning of the sentence. Two problems could arise: ambiguity at lower levels could swamp the system, leading to unacceptably high processing times; and disambiguation at higher levels could be too weak, letting too many final interpretations with equivalent ratings emerge. We investigate these problems by examining statistics from a set of test sentences.

The first statistic is a histogram of the test sentence lengths in words, shown in Figure 5. The LABEL column indicates the length of the sentence; the "#" column shows how many sentences had that length, and the PER and CUM columns are the percentage and cumulative percentage for the histogram variable. There were 158 sentences in the test set, which was a sampling of actual question asked of the data base. These sentences all had interpretations in D-LADDER. Note the small sample size for sentences of lengths 12,13,14, and 15; these points can be considered anomalous in the following graphs.

The parsing process introduces ambiguity by producing a number of alternative parsings for the same sentence. Figure 6 charts this ambiguity for the test set. This is a very important statistic because it measures the performance of the grammar in syntactic parsing. Note that over 50 percent of the sentences interpreted had only one parsing, or "construction." There are some sentences with large (>5) numbers of constructions; we need to do something about controlling this ambiguity. The next figure illustrates one source of the problem.

Figure 7 is a two-variable plot; the vertical variable is the sentence

27

```
LABEL                                          #    PER    CUM

  3 |***                                      5.0   3.2%   3.2%
  4 |********                                13.0   8.2%  11.4%
  5 |********                                13.0   8.2%  19.6%
  6 |****************                        25.0  15.8%  35.4%
  7 |********************                    30.0  19.0%  54.4%
  8 |**********************                  32.0  20.3%  74.7%
  9 |**********                              16.0  10.1%  84.8%
 10 |****                                     6.0   3.8%  88.6%
 11 |********                                13.0   8.2%  96.8%
 12 |*                                        2.0   1.3%  98.1%
 13 |*                                        2.0   1.3%  99.4%
 14 |*                                        1.0    .6%  100 %
 15 |                                         0.0   0.0%  100 %
                                           ----------------
                                          158.0  100 %
```

Mean is: 7.3

Figure 5: Histogram of Sentence Lengths in Words

```
LABEL                                          #    PER    CUM

      1 |**********************               86.0  54.4%  54.4%
      2 |**********                           39.0  24.7%  79.1%
      3 |***                                  11.0   7.0%  86.1%
      4 |*                                     6.0   3.8%  89.9%
      5 |**                                    7.0   4.4%  94.3%
      6 |                                      1.0    .6%  94.9%
      7 |                                      0.0   0.0%  94.9%
      8 |                                      1.0    .6%  95.6%
      9 |                                      1.0    .6%  96.2%
     10 |                                      1.0    .6%  96.8%
     11 |                                      0.0   0.0%  96.8%
     12 |                                      0.0   0.0%  96.8%
     13 |                                      1.0    .6%  97.5%
     14 |*                                     3.0   1.9%  99.4%
     15 |                                      0.0   0.0%  99.4%
 16 - 50 |                                     1.0    .6%  100 %
                                           ----------------
                                          158.0  100 %
```

Mean is: 2.4

Figure 6: Histogram of the Number of Constructions per Sentence

```
LABEL                                                    #

    3 |*                                                1.0
    4 |*                                                1.2
    5 |**                                               1.4
    6 |**                                               1.4
    7 |**                                               1.5
    8 |***                                              2.2
    9 |***                                              2.7
   10 |***                                              2.2
   11 |******                                           5.0
   12 |**************                                  11.5
   13 |********************                            16.5
   14 |******                                           5.0
   15 |                                                 0.0
```

Figure 7: Average Number of Constructions by Sentence Length

length and the horizontal variable the average number of constructions. The data point for sentences of length 14 is anomalous; there is only one sentence of length 14. The trend is for the number of constructions to rise exponentially with sentence length. This is because ambiguous syntax in a constituent has a multiplicative effect on the number of final constructions, e.g., a sentence with two different NPs, each with two syntactic interpretations, gives rise to 2x2=4 constructions. As sentence size grows, the number of ambiguous components tends to grow linearly, and thus the number of constructions for the whole sentence grows exponentially. This is a good argument for performing semantic and pragmatic processing before the entire sentence is assembled syntactically.

There are also a few ways in which the grammar can be changed to minimize syntactic ambiguity. Chief among these is attaching successive prepositional phrases as an unambiguous syntactic construction, and then letting the semantics determine where they go (note that intermixing semantics and syntax could achieve this purpose without changing the grammar). Sentences with the highest number of constructions tended to have multiple successive propositional phrases, e.g., "What is the steaming time from Norfolk to Gibraltar for the Sterett?" had 28 constructions.

How much does the introduction of domain semantics minimize syntactic ambiguity? Figure 8 enumerates translations that survive tests of domain

```
LABEL                                        #     PER    CUM

      1  |*********************          132.0   83.5%  83.5%
      2  |***                             23.0   14.6%  98.1%
      3  |                                 0.0    0.0%  98.1%
      4  |                                 1.0     .6%  98.7%
      5  |                                 1.0     .6%  99.4%
      6  |                                 1.0     .6%  100 %
      7  |                                 0.0    0.0%  100 %
      8  |                                 0.0    0.0%  100 %
      9  |                                 0.0    0.0%  100 %
 10 - 50  |                                0.0    0.0%  100 %
                                        ---------------
                                        158.0   100 %
```

Mean is: 1.2

Figure 8: Histogram of the Number of Translations per Sentence

acceptability. A "translation" of a sentence includes (for D-LADDER) finding referents to pronouns and filling in missing arguments to predicates (semantic ellipsis). Note that the translation process itself can introduce ambiguities; e.g., there may be more than one referent for a pronoun.

Altogether, this statistic is very encouraging, since the mean number of translations is 1.2, versus a 3.5 mean for the number of constructions. We discuss this issue more carefully at the end of this section, since most of the multiple translations actually have the same domain semantics.

The next three histograms give timing results for the interpretation process as a whole, and for the construction and translation processes separately. The LABEL column is the CP time in seconds for the given process; the # column lists the number of sentences for which the processing took LABEL amount of time. In Figure 9, for example, 78 sentences took between one and two seconds for a complete interpretation, which includes parsing the sentence (construction) and finding an interpretation of the parse tree within the domain (translation). SODA query generation times were not considered. Total parsing time in Figure 9 includes both construction and translation times. The mean time is a little over two seconds/sentence, which, while high, is not unreasonable. Almost 70 percent of the interpreted sentences took less than two seconds. The maximum parsing time for any sentence was a little over ten seconds.

30

```
LABEL                                          #      PER    CUM

0 - 1    |*******                            32.0   20.3%  20.3%
1 - 2    |***********************            78.0   49.4%  69.6%
2 - 3    |*****                              19.0   12.0%  81.6%
3 - 4    |****                               14.0    8.9%  90.5%
4 - 5    |*                                   3.0    1.9%  92.4%
5 - 6    |*                                   2.0    1.3%  93.7%
6 - 7    |*                                   5.0    3.2%  96.8%
7 - 8    |                                    0.0    0.0%  96.8%
8 - 9    |                                    1.0     .6%  97.5%
9 - 10   |*                                   2.0    1.3%  98.7%
10 - 50  |*                                   2.0    1.3%  100 %
                                            ----------------
                                            158.0   100 %
```

Mean is: 2.1 seconds

Figure 9: Histogram of Total Interpretation Times (seconds)

```
LABEL                                          #      PER    CUM

0 - .5    |******                            25.0   15.8%  15.8%
.5 - 1    |***********************           81.0   51.3%  67.1%
1 - 1.5   |******                            25.0   15.8%  82.9%
1.5 - 2   |***                               12.0    7.6%  90.5%
2 - 2.5   |*                                  3.0    1.9%  92.4%
2.5 - 3   |*                                  4.0    2.5%  94.9%
3 - 3.5   |*                                  5.0    3.2%  98.1%
3.5 - 4   |                                   0.0    0.0%  98.1%
4 - 4.5   |                                   1.0     .6%  98.7%
4.5 - 5   |                                   0.0    0.0%  98.7%
5 - 5.5   |                                   1.0     .6%  99.4%
5.5 - 6   |                                   1.0     .6%  100 %
6 - 50    |                                   0.0    0.0%  100 %
                                            ----------------
                                            158.0   100 %
```

Mean is: 1.06 seconds

Figure 10: Histogram of Construction Times (seconds)

31

```
LABEL                                   #     PER    CUM

0 - .5     |********************       66.0   41.8%  41.8%
.5 - 1     |**************             47.0   29.7%  71.5%
1 - 1.5    |*****                      18.0   11.4%  82.9%
1.5 - 2    |***                         9.0    5.7%  88.6%
2 - 2.5    |*                           4.0    2.5%  91.1%
2.5 - 3    |*                           4.0    2.5%  93.7%
3 - 3.5    |*                           4.0    2.5%  96.2%
3.5 - 4    |                            0.0    0.0%  96.2%
4 - 4.5    |                            1.0     .6%  96.8%
4.5 - 5    |*                           2.0    1.3%  98.1%
5 - 5.5    |                            1.0     .6%  98.7%
5.5 - 6    |                            0.0    0.0%  98.7%
6 - 10     |*                           2.0    1.3%  100 %
10 - 15    |                            0.0    0.0%  100 %
15 - 50    |                            0.0    0.0%  100 %
                                      ----------------
                                      158.0   100 %
```

Mean is: 1.03 seconds

Figure 11: Histogram of Translation Times (seconds)

As can be seen from Figures 10 and 11, translation and construction times are comparative; their means differ by only .03 seconds. Surprisingly, the translation time is not strictly a function of the number of constructions in a sentence; this can be seen by looking at the average translation time versus number of constructions, in Figure 12. The relative stability of translation times from 4 to 13 constructions indicates that, while the number of constructions is a factor in determining translation time, the semantic difficulty of the sentence is critical. The translation process is not spending a uniform amount of time on each construction; some are rejected relatively quickly, while others require substantial processing before they are accepted or rejected. This is a consequence of the deductive nature of the translation process -- finding an interpretation of a construction involves finding connection paths in the domain semantics, and some paths are more difficult than others to find.

The end result of ambiguity at the syntactic level is that the average parsing time tends to rise dramatically with sentence length, as shown by Figure 13. Although parsing times remain reasonable for sentences of length 10 and under, they begin to grow alarmingly at length 11. Recall from Figure 7 that the number of constructions begins to rise at the length 11 point. Controlling

32

```
LABEL                                                  #

      1  |*                                           .5
      2  |* * *                                      1.1
      3  |* * * *                                    1.3
      4  |* * * * *                                  1.6
      5  |* * * * * * * *                            2.6
      6  |* * * * *                                  1.5
      7  |                                           0.0
      8  |* * * * * * * * *                          3.2
      9  |* * * * * *                                2.1
     10  |* * * * * *                                2.0
     11  |                                           0.0
     12  |                                           0.0
     13  |* * * * * * * *                            2.9
     14  |* * * * * * * * * * * * * * * * *          6.6
     15  |                                           0.0
 16 - 50  |* * * * * * * * * * * * * *               5.3
```

Figure 12: Average Translation Time by Number of Constructions

```
LABEL                                                  #

      3  |* *                                         .8
      4  |* *                                         .7
      5  |* * *                                      1.2
      6  |* * *                                      1.4
      7  |* * * *                                    1.5
      8  |* * * * *                                  2.0
      9  |* * * * * * *                              3.1
     10  |* * * * *                                  2.2
     11  |* * * * * * * * * *                        4.5
     12  |* * * * * * * * * * * * * * * * * * *      8.4
     13  |* * * * * * * * * * * * * * * *            7.2
     14  |* * * * * * * * * * * * * * *              6.6
     15  |                                           0.0
```

Figure 13: Average Parsing Time by Sentence Length

syntactic ambiguity is crucial if we are to achieve reasonable parsing times.

Let us look at the translation process more closely. Whereas the number of constructions tended to grow exponentially with sentence length, the number of translations has a much more moderate growth, as shown in Figure 14. Disregarding the anomalous points 12,13, and 14, there is only a slight increase

33

LABEL   #

```
 3 |*****                                            1.0
 4 |*****                                            1.0
 5 |*****                                            1.0
 6 |******                                           1.1
 7 |******                                           1.2
 8 |******                                           1.2
 9 |*******                                          1.4
10 |*****                                            1.0
11 |********                                         1.6
12 |*******************                              4.0
13 |*****                                            1.0
14 |*****                                            1.0
15 |                                                 0.0
```

Figure 14: Average Number of Translations by Sentence Length

in the number of translations, from an average of 1.0 per sentence, to an average of 1.6 per sentence at length 11.

If we look at the sentences that had ambiguous translations, we can ask how many of those ambiguous translations had the same domain semantics, i.e., had the same conceptual schema instantiation. The next statistic shows that there were only 7 sentences with truly ambiguous semantics (Figure 15).

LABEL   #   PER   CUM

```
 2 |********************                        7.0   100 %  100 %
 3 |                                            0.0   0.0%   100 %
 4 |                                            0.0   0.0%   100 %
 5 |                                            0.0   0.0%   100 %
                                               ----------------
                                                7.0   100 %
```
        Mean is: 2.0

Figure 15: Histogram of Ambiguous Semantic Interpretations

Of these 7 sentences, all had only two semantic ambiguous interpretations. In order for a sentence to have a semantically ambiguous interpretation, it must have had two separate constructions, each of which translated preferentially to have a different semantics. The control structure of the inference processes currently used in the semantics routines is oriented toward keeping only the

34

best semantic interpretation; hence ambiguous semantics for the same construction, while considered by the semantics routines, are generally discarded by the time a sentence is fully interpreted.

The end result is that we have the histogram of Figure 16 for the number of semantic interpretations of sentences. The mean, 1.04, is substantially less than the 1.2 mean for the average number of translations per sentence.

```
LABEL                                    #      PER    CUM

  1 |*********************            149.0   95.5%  95.5%
  2 |*                                  7.0    4.5%  100 %
  3 |                                   0.0    0.0%  100 %
  4 |                                   0.0    0.0%  100 %
  5 |                                   0.0    0.0%  100 %
  6 |                                   0.0    0.0%  100 %
  7 |                                   0.0    0.0%  100 %
  8 |                                   0.0    0.0%  100 %
  9 |                                   0.0    0.0%  100 %
 10 |                                   0.0    0.0%  100 %
                                      -----------------
                                     156.0   100 %

Mean is: 1.04
```

Figure 16: Histogram of Number of Semantic Interpretations

Finally, we look at the score changes for semantically ambiguous sentences. The score of a sentence is a weighted combination of syntactic, semantic, and pragmatic factors that give a figure of merit for that sentence. In a semantically ambiguous situation, we would prefer the interpretation with a higher score. The scores range from 0 to 1000; 750 is the "median" score, in the sense that if no factors are introduced, a sentence will have this score. The histogram of Figure 17 indicates the scores of the best translation (i.e., for ambiguous sentences, the translation with the highest score) of a sentence. Note the peak at 700-800; this is caused by a large number of sentences that parse without exciting any scoring factors. We can now ask how the scores help to differentiate among the sentences with ambiguous semantics. Figure 18 illustrates the score differential between competing translations for those seven sentences that had truly ambiguous interpretations. The sample size is small, since only 7 sentences had ambiguous semantics. The mean separation is about 230. This is a large separation, and indicates that the scoring is doing a good

35

```
LABEL                                          #     PER    CUM

 0 - 100   |                                  0.0    0.0%   0.0%
100 - 200  |*****                            11.0    7.0%   7.0%
200 - 300  |*****                            11.0    7.0%  13.9%
300 - 400  |**********                       21.0   13.3%  27.2%
400 - 500  |*****************                34.0   21.5%  48.7%
500 - 600  |**********                       21.0   13.3%  62.0%
600 - 700  |*******                          15.0    9.5%  71.5%
700 - 800  |********************             41.0   25.9%  97.5%
800 - 900  |**                                4.0    2.5%  100 %
                                           ------------------
                                            158.0   100 %
```

Mean is: 529

Figure 17: Histogram of Scores of Best Translations

```
LABEL                                          #     PER    CUM

 0 - 100   |*************                     2.0   28.6%  28.6%
100 - 200  |                                  0.0    0.0%  28.6%
200 - 300  |*************                     2.0   28.6%  57.1%
300 - 400  |********************              3.0   42.9%  100 %
400 - 500  |                                  0.0    0.0%  100 %
500 - 600  |                                  0.0    0.0%  100 %
                                           ------------------
                                              7.0   100 %
```

Mean is: 228

Figure 18: Histogram of Score Changes
for Semantically Ambiguous Sentences

job in differentiating the interpretations. In all cases, the highest scoring
interpretation was the correct one.

The main conclusion we can draw from the statistics is that ambiguity at
the lower levels of processing, especially the syntactic level, is a serious
problem. Domain semantics can effectively reduce this ambiguity, but it needs
to be applied relatively early in the interpretation process if it is to be
effective in reducing interpretation times to reasonable levels.

2. Linguistic Coverage

The following are the results of an informal linguistic performance comparing D-LADDER and S-LADDER. We were mainly interested in checl the linguistic coverage of the grammar, and so chose a sentence set and r; the results on that basis. We were not performing a test of the concep closure of these systems, or the extent of the vocabulary, or any numbei other interesting parameters.

For simplicity, the test centered around a single concept--an off commands a ship. To introduce some syntactic variations, the additi concept of names for the officer and the ship were needed; these were no themselves the cause for failure of either system.

Although the vocabulary used was minimal, for the word OFFICER we to substitute COMMANDER when using the S-LADDER system. Although produced some awkward-sounding sentences, we felt it was only fair, as were not concerned with the particular vocabulary used.

We also tried to make each syntactic variation test meaningful, so t for instance, a failure in a previous syntactic test would not be the cause future failure. In short, we tried to make each sentence independent of *failure of previous sentences.*

Since we chose the syntactic variations, they represent an implicit *from our association with the grammar of D-LADDER.* But we did not use grammar in an explicit manner in making the variations; rather, we simply t to think up reasonable sentences to express the concepts in diffe grammatical ways. Of course, it would be best to garner test sentences actual users of the system.

The results are presented by listing the sentences used (by synt; variation category), and summarizing the results after each syntactic vari; category. A system was considered successful if it produced the approp semantics.

37

	D-LAD	LAD
QUERY: Who commands Kennedy?	y	y

NP variations

	D-LAD	LAD
1. What officer commands Kennedy?	y	y
2. Whose commander commands Kennedy?	y	n
3. Who commands the ship Kennedy?	y	y
4. Who commands the ship which is named Kennedy?	y	n
5. Who commands the ship whose name is Kennedy?	y	n
6. Who commands the ship of name Kennedy?	y	y
7. Who commands the ship which has the name of Kennedy?	y	y
8. Who commands the ship which has the name Kennedy?	y	n
9. Who commands the ship in the Med which has the name of Kennedy?	y	y
10. Who commands the ship in the Med of type DDG which has the name Kennedy?	y	n
11. The commander of the Kennedy commands what ship?	n	n
12. The commander of what ships commands the Kennedy?	n	n
13. What officer who commands ships in the Med commands Kennedy?	y	n
	11	6

PASSIVE variations

	D-LAD	LAD
14. Is Kennedy commanded by the commander of Lafayette?	y	n
15. By whom is Kennedy commanded?	y	y
	2	1

38

VERBAL variations

16. What is the name of the officer commanding Kennedy?   y n

17. What is the name of the ship commanded by
     the commander of Lafayette?   y n
                       ----
                      2 0

NOMINALIZATION variations

18. Who has command of the Kennedy?   y n

19. Who has Kennedy's command?   y n

20. To what officer does the command of
    the Kennedy belong?   y n
                       ----
                      3 0

The final score is: D-LADDER, 18 of 20
           S-LADDER, 7 of 20.

   These results satisfy our intuitive notions about the sensitivity of a semantic grammar to slight linguistic variations. The linguistically motivated grammar of D-LADDER succeeded much better in capturing the linguistic generalizations from which the variations were generated. The only failures occurred in sentences 11 and 12; this failure can be traced to the grammar, where wh-NPs are allowed only at the beginning of a sentence. To accept these sentences requires (in D-LADDER) writing another rule for wh-NPs, and adding another translator routine. In S-LADDER it would involve writing a new rule for every rule that uses a wh-word (and there are many).

E. Conclusions

   In an implementation of our proposed framework (called D-LADDER), we investigated a straightforward control strategy in which structures were built up sequentially from one level to the next, keeping all ambiguity at each level.

39

Experiments with a test set of 158 questions revealed that a strict separation of linguistic knowledge and domain semantics in the control strategy led to unacceptably high interpretation times for sentences that were syntactically highly ambiguous. We are currently investigating control strategies that allow domain-specific processing to be integrated more closely with early stages of linguistic processing, while still maintaining the structure of the transportable framework.

Finally, we note that the task of encoding a domain in a conceptual schema and semantic operators can be a difficult one. The framework itself does nothing to make this task easier; it requires that the interface designer provide some domain representation in which to evaluate the semantic operators. A current project at SRI called KLAUS is an investigation of methods to help a domain designer to quickly build a domain representation and deductive processes within it [8].

REFERENCES

1. G. Brown, "The Relational Model for the Blue File Data Base (Revised)," Naval Electronics Laboratory Center, San Diego, California (November 1976).

2. D. Bobrow and T. Winograd, "An Overview of KRL, A Knowledge Representation Language," Cognitive Science, Vol. 1, No. 1 (January 1977).

3. CODASYL Data Base Task Group, April 1971 Report, ACM, New York, 1971.

4. R. C. Moore, "Mechanical Intelligence: Research and Applications," Final Report, ARPA Contract No. N00039-78-C-0060, Artificial Intelligence Center, SRI International, Menlo Park, California (August 1979).

5. B. J. Grosz, "Utterance and Objective: Issues in Natural-Language Communication," Proc. 6th International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979).

6. E. D. Sacerdoti and D. Sagalowicz, "Ladder User's Guide," Artificial Intelligence Center, SRI International, Menlo Park, California (1979). (new version to be published)

7. J. J. Robinson, "DIAGRAM: an Extendable Grammar for Natural Language Dialogue," Artificial Intelligence Center, SRI International (1980). (paper in preparation).

8. G. G. Hendrix, "Research on Interactive Acquisition and Use of Knowledge," Research Proposal No. ECU 79-110, Artificial Intelligence Center, SRI International, Menlo Park, California (September 1979).

9. E. D. Sacerdoti, "Language Access to Distributed Data with Error Recovery," Proc. 5th International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts (August 1977).

10. B. H. Thompson and F. B. Thompson, "Rapidly Extendable Natural Language," Proc. of the ACM National Conference, pp. 173-182, (December 1978).

11. Richard Bisbey II and D. Hollingworth, "Situation Display: A Command and Control Graphics Application" (in progress)

12. R. E. Fikes and G. G. Hendrix, "A Network-Based Knowledge Representation and Its Natural Deduction System," Proc. 5th International Joint Conference on Artificial Intelligence, pp. 235-246, Cambridge, Massachusetts (August 1977).

13. R. C. Moore, "Handling Complex Queries in a Distributed Data Base," Technical Note 170, Artificial Intelligence Center, SRI International, Menlo Park, California (September 1979).

14. M. Stefik, "An Examination of a Frame-Structured Representation System," HPP-78-13, Stanford University, Stanford, California (September 1978).

15. G. G. Hendrix and W. H. Lewis, "TED: A Transportable English Data Manager," First Semiannual Report, SRI Project 7910, Artificial Intelligence Center, SRI International, Menlo Park, California (October 1979).